LAPORAN PRAKTIKUM
MESSAGE PASSING INTERFACE (MPI)

DISUSUN OLEH :

NAMA : SISKIA ISRAWANA

NIM : 09011282227099

KELAS : SK3C INDRALAYA

DOSEN PENGAMPU

Adi Hermansyah, S.Kom., M.T.

PROGRAM STUDI SISTEM KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SRIWIJAYA

2023

Perbandingan kecepatan program yang dijalankan secara serial dan paralel menggunakan MPI, terdapat 2 program yaitu :

1. siskia1.py (bubble sort)

```python
from siskia1 import MPI
import siskia1 as np
import time

def parallel_bubble_sort(data, comm):
    rank = comm.Get_rank()
    size = comm.Get_size()

    local_data = np.array_split(data, size)[rank]

    n = len(local_data)
    for i in range(n):
        for j in range(0, n - i - 1):
            if local_data[j] > local_data[j + 1]:
                local_data[j], local_data[j + 1] = local_data[j + 1], local_data[j]

    sorted_data = comm.gather(local_data, root=0)

    if rank == 0:
        merged_data = np.concatenate(sorted_data)
        merged_data.sort()
        return merged_data
    else:
        return None

if __name__ == "_main_":
    comm = MPI.COMM_WORLD
    rank = comm.Get_rank()

    if rank == 0:
        num_elements = 5
        data = np.random.randint(0, 100000000, num_elements)

    else:
        data = None

    data = comm.bcast(data, root=0)

    start_time = time.time()
    sorted_data = parallel_bubble_sort(data, comm)
    end_time = time.time()

    if rank == 0:
        print("Array yang diurutkan:", sorted_data)
        elapsed_time = end_time - start_time
        print(f"Waktu pemrosesan: {elapsed_time:.4f} detik")
```

2. siskia2.py (numerik)

```python
from siskia2 import MPI
import siskia2 as np
import time

comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

def generate_random_matrix(rows, cols):
    return np.random.rand(rows, cols)

def main():
    ordo = 3

    baris = ordo
    kolom = ordo + 1

    A = generate_random_matrix(baris, kolom)

    start_time = time.time()

    for i in range(baris):
        diag = A[i][i]
        for j in range(kolom):
            A[i][j] /= diag
        for k in range(i + 1, baris):
            diag1 = A[k][i]
            for j in range(0, kolom):
                A[k][j] = A[k][j] - diag1 * A[i][j]

    comm.barrier()

    if rank == 0:
        print("Eliminasi Gauss:")
        for i in range(min(5, baris)):
            for j in range(kolom):
                print(f"{A[i][j]:.2f}", end=" ")
            print("...")

        x = np.zeros(ordo, dtype=float)
        for i in range(ordo - 1, -1, -1):
            x[i] = A[i][kolom - 1]
            for j in range(i + 1, ordo):
                x[i] -= A[i][j] * x[j]

        end_time = time.time()
        elapsed_time = end_time - start_time

        for i in range(min(5, ordo)):
            print(f"x{i + 1} = {x[i]:.2f}")

        print(f"Estimasi Waktu proses: {elapsed_time:.6f} detik")

if __name__ == '_main_':
    main()
```

Agar program dapat berjalan dengan stabil untuk menghitung SPL (sistem persamaan linier) menggunakan metode eliminasi gauss. Dan Program akan dijalankan pada3 VM menggunakan *Host Only Adapter*.

A. Program Pertama

Contoh output program "siskia1.py"

```
zorin@master:~$ python3 /home/python/coding.py
Array yang diurutkan: [39755316 43999578 69484001 70604163 87345563]
Waktu pemrosesan: 0.0001 detik
zorin@master:~$
```

Tabel Prakrikum program "siskia1.py"

Jumlah Data	Estimasi Waktu (detik)	
	Serial	Paralel
10	0,0001	0,0012
100	0,0012	0,0056
1.000	0,1093	0,0215
10.000	11,0147	1,1391

Pada **program "siskia1.py"** dengan ukuran data dari 10, 100, 1.000 dan 10.000 data. Jika program berhasil maka akan menghasilkan keluaran berupa bilangan acak yang diurutkan mulai dari bilangan terkecil hingga terbesar dan diikuti dengan stopwatch sebagai acuan. Program akan dijalankan dengan 2 kondisi kontrol.Perintah pertama adalah "python3 nama_program" untuk menjalankan program dengan 1 VM dan "mpiexec -n jumla_komputer -host, nama_host1 dan host selanjutnya python3 nama_program" untuk menjalankannya secara paralel.

B. Program Kedua

Contoh output program "siskia2.py"

```
zorin@master:~$ python3 /home/python/dica.py
Eliminasi Gauss:
1.00 4.20 2.98 0.61 ...
-0.00 1.00 0.58 -0.13 ...
-0.00 -0.00 1.00 2.31 ...
x1 = -0.14
x2 = -1.46
x3 = 2.31
Estimasi Waktu proses: 0.000154 detik
```

Tabel Prakrikum program "siskia2.py"

Jumlah Ordo	Estimasi Waktu (detik)	
	Serial	Paralel
100	0,236940	0,257183
200	1,986445	1,799140
300	6,150870	5,961488
400	15,701769	14,182887
500	32.065418	27,785337

Pada program kedua, kita akan menggunakan program eliminasi Gauss yang dimodifikasi yang dapat dijalankan secara paralel menggunakan MPI dengan program "siskia2.py. Jika berhasil maka program akan mengeluarkan output berupa perhitungan SPL dengan menggunakan eliminasi Gauss berdasarkan jumlah data program yang berhubungan dengan timer. Sama seperti program pertama, program kedua akan berjalan secara seri dan paralel dengan ordo data 100, 200, 300, 400 dan 500. Di ats adalah hasil benchmark dari program kedua.

C. Analisa da Kesimpulan

Pertama kita perlu menentukan dan melihat berapa banyak data yang akan diproses sebelum memutuskan apakah akan menggunakan paralel atau serial. Salah satu solusinya adalah dengan menentukan "batas rata-rata" di mana kesenjangan antara perkiraan waktu paralel dan serial tidak berbeda secara signifikan.Pemrosesan paralel akan sangat cocok digunakan pada kondisi dimana jumlah data yang dikirim sangat banyak. Pada saat yang sama, pemrosesan serial menghemat waktu pengguna untuk jumlah data masuk yang relatif kecil.

Berdasarkan tabel pertama dan kedua, jumlah data dapat sangat mempengaruhi kecepatan dimulainya pemrosesan data. Jika Anda memproses data dalam jumlah besar, pemrosesan paralel lebih efisien karena perkiraan waktu yang dibutuhkan lebih sedikit dibandingkan pemrosesan serial. Hal ini karena pemrosesan paralel berbagi tugas dengan mesin virtual lainnya, sehingga beban dibagi secara merata dan waktu yang terbuang oleh pemrosesan serial dapat dikurangi. Terlihat bahwa ketika jumlah data pada program pertama 1.000 dan 10.000 terlihat kecepatan pemrosesan paralel jauh melebihi kecepatan pemrosesan data serial. Begitu pula dengan jumlah pesanan dari 300-500 pada program kedua. Semakin besar jumlah data yang diproses, semakin besar kesenjangan antara perkiraan waktu serial dan paralel. Namun hal yang sama akan terjadi sebaliknya. Semakin kecil jumlah data, semakin lambat pemrosesan paralel dibandingkan dengan pemrosesan serial.Hal ini karena pemrosesan paralel membutuhkan lebih banyak waktu untuk mentransfer data dan mengirim hasilnya kembali ke mesin virtual lainnya.Menjadikan pemrosesan serial mampu menampilkan output lebih cepat karena tidak perlu mendistribusikan bebannya ke mesin virtual lain.Misalnya sekitar 100 hingga 200 pada program kedua dan data 10 hingga 100 pada program pertama.